Hardware features:

VMMR, the VME-receiver module

The VME receiver Module VMMR-8 / 16 has 8 or 16 optical buses.
With 16 buses, 8 of them are back on the motherboard.

The module synchronises the front end clocks, receives triggers
on the buses and accepts external gates.
In specified coincidence cases the events are requested from the buses
processed to a common event and store in a large memory.
There it is available for VME readout. The event format is very similar
to other mesytec modules like MDPP-16.
Only the address space for more channels is larger.

The 16 bus module can receive up to 2048 channels.
The number of VMMR VME modules in a setup is unlimited.

Optical Bus

POF twin fiber to transmit up to 200 Mbit/s of data.
Diameter of light guide = 1 mm, jacket diameter 2.2 mm.
LED transmitter/receiver: 650 nm (red).

- Live insertion (can be inserted in a running crate)
- Power consumption: 6.5 W,
  +5 V, 1.2 A
  -12 V, 50 mA

ATTENTION !!

General for VME64X modules:
The locker screws of of the modules must be fixed for operation.

mesytec GmbH & Co. KG
Wernher-von-Braun-Str. 1, D-85640 Putzbrunn, Germany
phone: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

Control input / output

- Differential control inputs:
  - interface any differential signals: ECL, LVDS or LVPECL. They can be individually
  - terminated (110 Ω) via register settings

- NIM inputs:
  - standard NIM, 50 Ω

- NIM output:
  - -0.7 V when terminated with 50 Ω

- mesytec control bus output, shares connector with busy output. +0.7 V terminated

Minimum trigger width for individual inputs is = 10 ns
Maximum external reference synchronization clock frequency (sync input): 75 MHz

VMMR-16 register set, Data FIFO, read data at address 0x0000

(access R/W D32, 64)
only even numbers of 32 bit-words will be transmitted. In case of odd number of data words, the last word will be a fill word (= 0).
FIFO size: 64k - 4k words with 32 bit length

Header (4 byte)

2 header signature	2 subheader	4	8 module id	4	12 number of following data words, including EOE
b01	b00	b000,trig	module id	0x0	number of 32 bit data words

Data (4 byte) DATA event

2 data-sig	2	4	8	16
b00	11	Bus Number	0x00	Time difference Gate start to bus trigger

Data (4 byte) DATA event

2 data-sig	2	4	12	12
b00	01	Bus Number	Front end subaddress	ADC value

Data (4 byte) Extended time stamp

2 data-sig	2	12	16
b00	10	xxxx xxxx xxxx	16 high bits of time stamp

Data (4 byte), fill dummy (to fill MBLT64 word at odd data number)

2 data-sig	30
b00	0

End of Event marker (4 byte)

2	30
b11	event counter / time stamp

Registers, Starting at address x6000 (access D16)

Address	Name	Bits	dir	default	Comment
	Address registers				
0x6000	address_source	1	RW	0	0 = from board coder, 1 from address_reg
0x6002	address_reg	16	RW	0	address to override decoder on board
0x6004	module_id	8	RW	0xFF	is part of data header If value = FF, the 8 high bits of base address are used (always board coder).
0x6006	Fast MBLT	1	W	1	0 = reduce MBLT cycle time to 100ns
0x6008	soft_reset	1	RW		breaks all activities, sets critical parameters to default. **Wait 200 ms after writing this register.** read: Hardware index: 0x5006
0x600E	firmware_revision	16	R		Example: rev 1.4 = 0x0104

	IRQ (ROACK)				
0x6010	irq_level	3	RW	0	IRQ priority 1...7, 0 = IRQ off
0x6012	irq_vector	8	RW	0	IRQ return value
0x6014	irq_test	0	W		initiates an IRQ (for test)
0x6016	irq_reset	0	W		resets IRQ (for test)
0x6018	irq_data_threshold	15	RW	1	Every time the number of 32 bit words in the FIFO exceeds this threshold, an IRQ is emitted. Maximum allowed threshold is "FIFO size".
0x601A	Max_transfer_data	15	RW	1	1) Specifies the amount of **data** read from FIFO before Berr is emitted. Only active for multi **event mode 3**. Transfer is stopped only after full events. Example: At Max_transfer_data = 1, 1 event per transfer is emitted. 2) Specifies the number of **events** read from FIFO before Berr is emitted. Active for multi **event mode 0xb.** Setting the value to 0 allows unlimited transfer.
0x601C	IRQ_source	1	RW	1	IRQ source: 0 = **event** threshold exceeded 1 = **data** threshold exceeded
0x601E	irq_event_threshold	15	RW	1	Every time the number of events in the FIFO exceeds this threshold, an IRQ is emitted.

For multi event mode 2 and 3 the IRQ is:

- **set** when the FIFO fill level gets more than the threshold and is
- **withdrawn** when IRQ is acknowledged or when the fill level goes below the threshold

	MCST CBLT				
0x6020	cblt_mcst_control	8	RW	0	see table
0x6022	cblt_address	8	RW	0xAA	A31...A25 CBLT- address
0x6024	mcst_address	8	R	0xBB	A31...A25 MCST- address

Bit	Name	Write	Read
7	MCSTENB	1 Enable MCST 0 No effect	0
6	MCSTDIS	1 Disable MCST 0 No effect	1 MCST enabled 0 MCST disabled
5	FIRSTENB	1 Enable first module in a CBLT chain 0 No effect	0
4	FIRSTDIS	1 Disable first module in a CBLT chain 0 No effect	1 First module in a CBLT chain 0 Not first module in a CBLT chain
3	LASTENB	1 Enable last module in an CBLT chain 0 No effect	0
2	LASTDIS	1 Disable last module in an CBLT chain 0 No effect	1 Last module in a CBLT chain 0 Not last module in a CBLT chain
1	CBLTENB	1 Enable CBLT 0 No effect	0
0	CBLTDIS	1 Disable CBLT 0 No effect	1 CBLT enabled 0 CBLT disabled

CBLT Address Field

A31.....A24	A23.....A00
CBLT ADRS	8 high bits, not significant + 16 bit module address space

MCST Address Field

A31.....A24	A23.....A00
MCST ADRS	8 high bits, not significant + 16 bit module address space

At BLT32

When an empty module is accessed at address 0, BERR is emitted.

At CBLT

When no module contains data, no data are transmitted. The last module emits BERR.

	FIFO handling				
0x6030	buffer_data_length	16	R		amount of data in FIFO (only fully converted events). Units → data_len_format. Can be used for single- and multi event transfer
0x6032	data_len_format	2	RW	2	0 = 8 bit, 1 = 16 bit, 2 = 32 bit, 3 = 64 bit, 4 = show number of events in FIFO. The number of 32 bit words is always even. If necessary the fill word „0" is added. For len 0 and 1 the max value 0xFFFF is shown when number exceeds the 16 bit format. The FIFO is not affected.
0x6034	readout_reset		W		At single event mode (multi event = **0**): allow new trigger, allow IRQ At multi event = **1**: checks threshold, sets IRQ when enough data. Allows safe operation when buffer fill level does not go below the data threshold at readout. At multievent = **3**: clears Berr, allows next readout
0x6036	multi event	4	RW	0	<table><tr><th>Bit [3]</th><th>Bit [2]</th><th>Bit [1:0]</th></tr><tr><td>count events not words (reg. 0x601A)</td><td>skip berr, send EOB</td><td>**mode** [1:0]</td></tr></table> Allow multi event buffering (bit 0, 1) **mode = 0** → **no** (0x6034 clears event, allows new conversion) **mode = 1** → **yes**, unlimited transfer, no readout reset required (0x6034 can be written after block readout). Don´t use for CBLT **mode = 3** → **yes** but VMMR transfers limited amount of data. With reg 0x601A the number of data words can be specified. After word limits is reached, the next end of event mark terminates transfer by emitting Berr. So 0x601A = 1 means event by event transfer (Berr after each event). The next data block can be transferred after writing 0x6034 (resets Berr). **Berr handling:** when bit [2] is set: Send EOB = bit [31:30] = bx10 instead of Berr Bit [3]: Compare number of transmitted events (not words!) with max_transfer_data (0x601A) for Berr condition.
0x6038	marking_type	2	RW	0	00 → event counter 01 → time stamp 11 → extended time stamp → next page

0x603A	start_acq	1	RW	1	1 → start accepting triggers If no external trigger logic, which stops the gates when daq is not running, is implemented, this register should be set to 0 before applying the FIFO_reset to get a well defined status. When setting it to 1 again for data acquisition start, the module is in a well defined status.
0x603C	FIFO_reset		W		Initialize FIFO
0x603E	data_ready	1	R		1 → data available

	operation mode				
0x6040	bus_ok	16	R		At acquisition start, VMMR evaluates buses if they are connected. Result in binary representation: For example 0b0000000000000011, bus0,1 are connected.
0x6042	active_buses	16	RW	0xFFFF	a 1 means bus active, read: activated buses at start_daq
0x6046	Timing resolution (SW-RevX110 and higher)	1	RW	0	Set timing resolution 5ns = 0; 1ns = 1. 1ns setting requires MMR mudules with 1ns resolution. Trigger input with 1ns res., trigger out 5ns res.
0x604C	Gate_blocking_time	11	RW	0	multiples of 10ns; Gate can be blocked for this time.

	Trigger				
0x6050	win_start	13	RW	0x1000 -32	Unit: 5 ns Start window of interest: 0x0000 start at -20.48 us 0x1FFF start at +20.48 us 0x1000 = 4k, no delay < 4k, window starts before Trigger > 4k, window is delayed
0x6054	win_width	12	RW	64	Unit: 5 ns, max 4k = 20.48 us
0x6058	ext_trig_source	2	RW	3	trig1 / trig0 to trigger gate
0x605A	trig_source	16	RW	0	this register allows to set any number of individual MMR-Frontends (buses) creating the trigger. Bit 0 corresponds to bus 0, bit 15 to bus 15. A "1" marks the buses as trigger source.
0x605C	out_trigger_source	1	RW	0	0 = free trigger to trigger output (standard operation) 1 = output triggers which are accepted by window of interest (can be useful when self triggered and MVLC is the only trigger source)
0x605E	trigger_output	8	RW	0x00	If 0x605C = 0 (selects free trigger): Bit 0 corresponds to bus 0, bit 15 to bus 15. A "1" marks the bus as trigger source.

IO 0x6060	Inputs, outputs				
0x6060	ECL3	8	RW	0x00	INPUT lower 4 bit: 0 = Off, 1 = Trig0 in higher 4 bit: 0 = terminated, 1 = unterminated
0x6062	ECL2	8	RW	0x00	INPUT lower 4 bit: 0 = Off, 1 = Sync_in, 2 = Trig1 in higher 4 bit: 0 = terminated, 1 = unterminated **when sync is selected also set reg 0x6096 !!**
0x6064	ECL1	8	RW	0x00	INPUT lower 4 bit: 0 = Off, 1 = Reset_in higher 4 bit: 0 = terminated, 1 = unterminated
0x6066	ECL0	4	RW	0	OUTPUT 0 = Off, 4 = Busy, 8 = data in buffer above threshold 0x6018 (= Data ready) 9 = events in buffer above threshold 0x601E
0x6068	NIM3	2	RW	1	INPUT 0 = Off, 1 = Trig0_in, 2 = sync_in **when sync is selected also set reg 0x6096 !!**
0x606A	NIM2	2	RW	1	0 = off, 1 = Trig1_in, 2 = timestamp_reset
0x606C	NIM1	4	xx	x	trig_out
0x606E	NIM0				0 = Off, 1 = Cbus, 4 = Busy_out (= FIFO full or ACQ stopped) 8 = data in buffer above threshold 0x6018 9 = events in buffer above threshold 0x601E

The test pulses are generated in MMR frontends and are injected at the inputs of the preamps. They are initiated by a ping signal on the buses which is centrally created by the VMMR with a frequency of 6kHz. So pulses are created absolutely synchronous in all connected MMRs.
When “only triggers” is set, the MMRs start conversion without injecting a pulser signal. So only noise is converted.

0x6070	Test pulser				
0x6070	pulser_status;	1	RW	0	0 = Off, 1 = Pulser on (6kHz), (3 = send triggers only → read pedestals)
0x6072	pulser_amplitude	8	RW	100	maximum amplitude: 0xFF = 255

IO selection Overview

The numbers in the table can be written to the corresponding register.
For example: ECL2 should input the trigger 1: write 2 to 0x6062.

IO	default	TR0	TR1	SYN	RES	TRout	Busy	Rdy data	Rdy event	Cbus
ECL3	0	1								
ECL2	0		2	1						
ECL1	0				1					
ECL0	0						4	8	9	
NIM3	1	1		2						
NIM2	1		1		2					
NIM1	1					1				
NIM0	1						4	8	9	1

Selection of 0 always means it is unused or "Off"

Description

TR0 = Trigger 0 input
TR1 = Trigger 1 input
SYN = external Frequency to synchronize event time stamp
RES = reset for event time stamp
TRout = trigger output
Busy = module not ready to take more triggers
Rdy dat = data in buffer above threshold register "0x6018"
Rdy eve = events in buffer above threshold register "0x601E"
Cbus = control bus to control external mesytec modules (MHV-4, MPRB-32...)

Mesytec control bus:

MRC 0x6080	Module RC				
0x6080	rc_busno	2	RW	0	0 is external bus, comes out at busy output
0x6082	rc_modnum	4	RW	0	0...15 (module ID set with hex coder at external module)
0x6084	rc_opcode	7	RW		3 = RC_on, 4 = RC_off, 6 = read_id, 16 = write_data, 18 = read_data
0x6086	rc_adr	8	RW		module internal address, see box below
0x6088	rc_dat	16	RW		data (send or receive),write starts sending
0x608A	send return status	4	R		bit0 = active bit1 = address collision bit2 = no response from bus (no valid address)

Send time is 400 us. Wait that fixed time before reading response or sending new data.
Also polling at 0x608A for bit0 = 0 is possible
The Trigger0-LED shows data traffic on the bus, the Trigger1-LED shows bus errors
(i.e. non terminated lines)

Example for controlling external modules with mesytec RC-bus

Initialize and read out a MCFD-16 CFD- module.
MCFD-16 ID-coder set to 7
Bus line must be terminated at the far end.

Activate VMMR control bus at busy line

Write(16) addr 0x6074 data 1

Get Module ID-Code (= Type of module = 26 for MCFD-16)

```
Write(16)    addr 0x6082  data 7     // address module 7
Write(16)    addr 0x6084  data 6     // send code "read IDC"
Write(16)    addr 0x6088  data 0     // initialize send request. Data has no effect

Wait loop: Read(16) 0x608A  and compare bit0 to get 0.  Then evaluate other bits for error status

Read(16)     addr 0x6088  data 40    // at ID readout the bit 0 shows the module RC status
                                     // (1 is on). Bit 1..7 show the IDC
                                     // → interpretation: Module off, IDC = 20
```

Set threshold for channel 0 to 10

```
Write(16)    addr 0x6082  data 7     // address module 7
Write(16)    addr 0x6084  data 16    // code "write_data"
Write(16)    addr 0x6086  data 0     // address module memory location 1
Write(16)    addr 0x6088  data 10    // start send. Data to send

Wait loop: Read(16) 0x608A  and compare bit0 to get 0. Then evaluate other bits for error status.

Optional the read back data is available
Read(16)     addr 0x6088  data 10    // read back written data for control
```

Read threshold of channel 0

Write(16) addr 0x6082 data 7 // address module 7
Write(16) addr 0x6084 data 18 // code “read_data”
Write(16) addr 0x6086 data 0 // address module memory location 1
Write(16) addr 0x6088 data 0 // send read request. Data has no effect

Wait loop: Read(16) 0x608A and compare bit0 to get 0. Then evaluate other bits for error status

Read(16) addr 0x6088 data 10 // read out data, “10” returned

Activate RC in module

All set data will get active. This can also be done before setting the values.

Write(16) addr 0x6082 data 7 // address module 7
Write(16) addr 0x6084 data 3 // send code “RC_on”
Write(16) addr 0x6088 data 0 // initialize send request. Data has no effect

Deactivate VMMR-16 control bus at busy line

Write(16) addr 0x606E data 0 // busy output used as busy

CTRA

Time stamp counters, event counters
All counters have to be read in the order: low word then high word !!!
They are latched at low word read. The event counter counts events which are written to the buffer.

CTRA 0x6090	counters A				
0x6090	Reset_ctr_ab	2	RW		b0001 resets all counters in CTRA, b0010 resets all counters in CTRB, b1100 allows single shot reset for CTRA with first edge of external reset signal. the bit bx1xx is reset with this first edge Reset of "counters A" will also reset the global 46 bit TDC time stamp
0x6092	evctr_lo	16	R	0	event counter low value
0x6094	evctr_hi	16	R	0	event counter high value
0x6096	ts_sources	5	RW	b00	bit0: frequency source (VME = 0, external = 1) bit1: external reset enable = 1 bit4: CTRB "time" counts trigger outputs (= free triggers, selected by 0x605E)
0x6098	ts_divisor	16	RW	1	time stamp = time / (ts_divisor) 0 means division by 65536
0x609C	ts_counter_lo	16	R		Time low value
0x609E	ts_counter_hi	16	R		Time high value

CTRB

Counters are latched when VME is reading the low word
Output value is divided by 40 to give a 1 us time basis

CTRB 0x60A0	counters B				
0x60A8	time_0	16	R		Time [1 us] (48 bit) RCP: alternately: free triggers, selected by 0x6096 bit4, and 0x605E
0x60AA	time_1	16	R		
0x60AC	time_2	16	R		
0x60AE	stop_ctr	2	RW	0	0 = run, 1 = stop counter bit 0 all counter B bit 1 time stamp counter (A)

Bus addressing

0x6100	select_Bus	4	RW	0	0...15 selects individual buses, 16 selects all (only write to bus is possible)

Address 0x6110	**Parameter**			**default**	
6110	fe_addr_wr	8	RW	0	Set Write subaddress in Frontend
6112	fe_data_wr	16	W		Data write register, **starts write**. Write needs up to 650ns, independent of 1 or all buses write. buffer FIFO 32 Words
6114	fe_addr_rd	8	W		Read subadress data in Frontend, **starts read**. Result is written into an output FIFO (delay 2 us); Fifo depth 32 words
6116	Read_Address	8	R		read FIFO: Frontend: {4'h0,bus[3:0],Address [7:0]} Empty FIFO returns 0xFFFF
6118	Read_Data	16	R		read FIFO Frontend Data [15:0], and release next data. Empty FIFO returns 0xFFFF
611A	bus_delay	9	R	50	longest bus in [m], 0...50 allowed. 50m always works.

Frontend module registers

0x01	reset	0	W	resets front end data
0x03	LEDs_on	1	W	onboard LEDs activated (switch off for light sensitive detectors)
0x04	start	1	W	accept trigger, stop also clears buffers
0x05	temp	12		multiples of 0.1 °C
0x06	voltage	12		primary voltage, multiples of 100 mV
0x07	receive_power	12		Amount of received light on optical bus (arbitrary units)
0x08	Pulser_Ampl	8		DACA: pulser amplitude
0x09	com_thr0	8		DACB, trigger threshold, usually lower 32 channels, assigned application specific.
0x0A	com_thr1	8		DACC, trigger threshold, usually higher 32 channels, assigned application specific.
0x0B	aux_thr1	8		DACD, trigger threshold, usually 64 high channels for MMR128, assigned application specific.
0x0C	Pulser_ping	2	00	**Usually use central pulser (reg 0x6070), leave this register at default!**; default setting b00 = off b00 → one pulser ping; b01 → one trigger ping; b10 → periodic pulser; b11 → periodic trigger; Local pulser 1.2kHz;
0x10	data_threshold	12		For all MMR-channels, data below are suppressed
0x21	peak_delay	8		maximum possible delay between trigger and shaper peak Multiples of 10ns
0x22	active_bank	16		Factory setting, do not write. {tp3[3:0],tp2[3:0],tp1[3:0],tp0[3:0]} def: 0x0021 -> lower two banks; 0x8421 → all 4 banks
0x23	bus_error	16		bit errors occured on optical bus. Write: reset 0x23, 0x24

0x24	PLL Error	16		Desynchronisation count of PLL;
0x30	Firmware_revision	16		0x0006, 0x0007... Hex notation!
0x31	Hardware Revision	16		0x0014, 0x0015... Hex notation!

Read a register value from MMR connected to optical bus:
Request data
0x6100 Bus_Number // select the bus number to request data
0x6114 Register_address // select the register number in MMR module
0x6114 Register_address
.
Repeat Register request maximum 25 times, also different buses are possible
the request can also be sent to all buses by selecting bus number = 16. This initiates up to 16 registers to be read out. So for an VMMR-16 with all buses occupied it results in reading 16 registers.

Wait at least 1ms

Get data from FIFO (depth max 30 register contents). The requsted data are now in a FIFO:
0x6116 {bus_number[3:0], reg_Address[7:0]}
0x6118 Data[15:0] gets register content data from FIFO, **requests next data from FIFO**

Write a register value to MMR connected to optical bus:
16 cmds per bus, then wait some us. Bus address 16 “to all” cmds may be sent.

0x6100 Bus_Number[3:0] // to bus 0..15 or to all = 16
0x6110 Register addressing[7:0]
0x6112 Data to write[15:0] and **start send**

Special Trigger outputs

Starting with March 2021 a new feature is built into all firmware. The VME IRQ lines are mostly unused, but are very good high quality lines to the VME controller. A combination of input triggers can be sent to the 7 available IRQ lines, and can be processed in the Trigger IO of the MVLC controller. A collision of signals with other IRQ signaling of the module should be avoided. Several modules may control the same IRQ line, the outputs are then a wired OR of all participating modules. Pulse length is fixed 50ns.

VME-Addr	**Name**	Width	Direction	Value	Description
0x6300	**Trig to IRQ1**	16	RW	0x0000	Connect an OR of the selected bus triggers (up to 16) to IRQ line 1. Example 0b00000000 10000001 means bus 0 and 7 are Ored and sent to IRQ 1
0x6304	**Trig to IRQ2**	16	RW	0x0000	
0x6308	**Trig to IRQ3**	16	RW	0x0000	
0x630C	**Trig to IRQ4**	16	RW	0x0000	
0x6310	**Trig to IRQ5**	16	RW	0x0000	
0x6314	**Trig to IRQ6**	16	RW	0x0000	
0x6318	**Trig to IRQ7**	16	RW	0x0000	

FAQ:

*** Why there are two thresholds ?**

If any channel of one MMR gets an amplitude above the **discriminator threshold**, all channels are converted. Then the **data threshold** can be used to cut off noise (zeros) to reduce the amount of transmitted data.

*** How to adjust the thresholds.**

Usually adjust the threshold up until there are very few triggers per second.
* The triggers can be seen at the green LEDs of the VMMR buses.

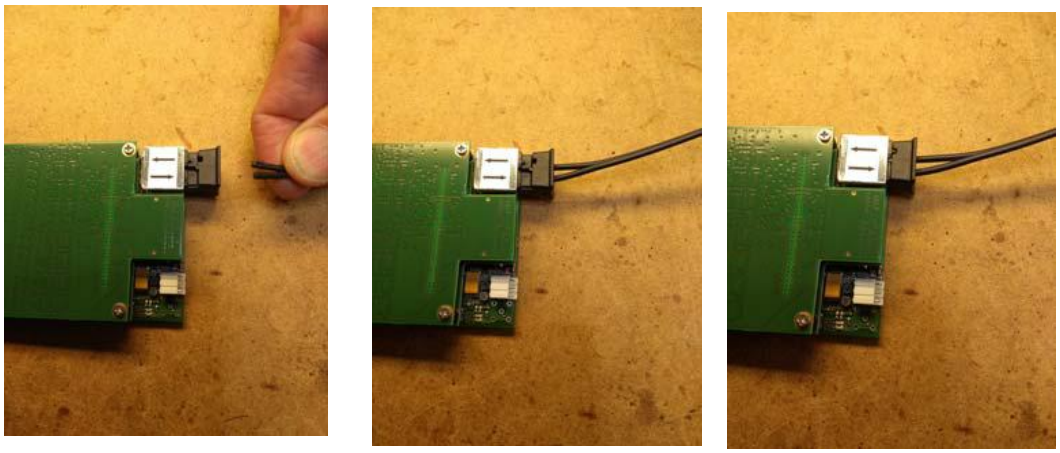
If the trigger rate has to be higher than can be checked by LED:
* set the trigger output to one bus and check with oscilloscope
* or check the rate of buses in the data acquisition.

For applications which require lowest threshold, some kHz of rate could be acceptable when a coincidence with the window of interest, triggered by an external low rate detector, can be established, or a very fast data acquisition allows readout of the large amount of data.

*** How the optical bus is connected:**

Connect the Fibers to VMMR module. Make a power cycle to let all optical transceivers get active.
At the other end of the fiber pairs you can now see light at one fiber.

Open the locker at MMR transceiver



MMR modulee with opened and closed locker

The fiber with red light has to be at the marked input (input arrow).
Insert the fiber pair into the transceiver (23 mm !!) until it hast contact to the internal optical units.
Some times with new cables it clamps somewhere inside with its jacket. So check carefully if it can

be inserted 23mm.
Close the locker to fix the fiber.
When the link is established, the red LED on the VMMR-bus should go off.